Database Replication The Easy Way

ITDevCon

Delphi European Conference

Raphael & Jonathan Neve
Microtec Informatique

October 27/28 2011 VERONA

SESSION OVERVIEW

- Approaches to data sync
- Real-world situations
- Tips in implementing replication
- Introducing CopyCat
- Hands-on demo

APPROACHES TO DATA SYNC

1) **Full database copy**

- Quick & dirty approach
- OK for occasional read-only access
- Not scaleable
- Not flexible
- Not optimal

APPROACHES TO DATA SYNC

2) **Selective data pumping**

- Valid for tables that are small or quite static
- Simple and reliable
- Can mix & match other methods for larger or more dynamic tables
- Still not optimal resource usage
- Still read-only !

APPROACHES TO DATA SYNC

3) **Change flagging**

- Uses a field to mark records that have changed
- Better resource management
- Finer control than the previous methods
- Doesn't allow multiple node sync
- Can be complex & error prone
- Potentially read/write but complex to handle

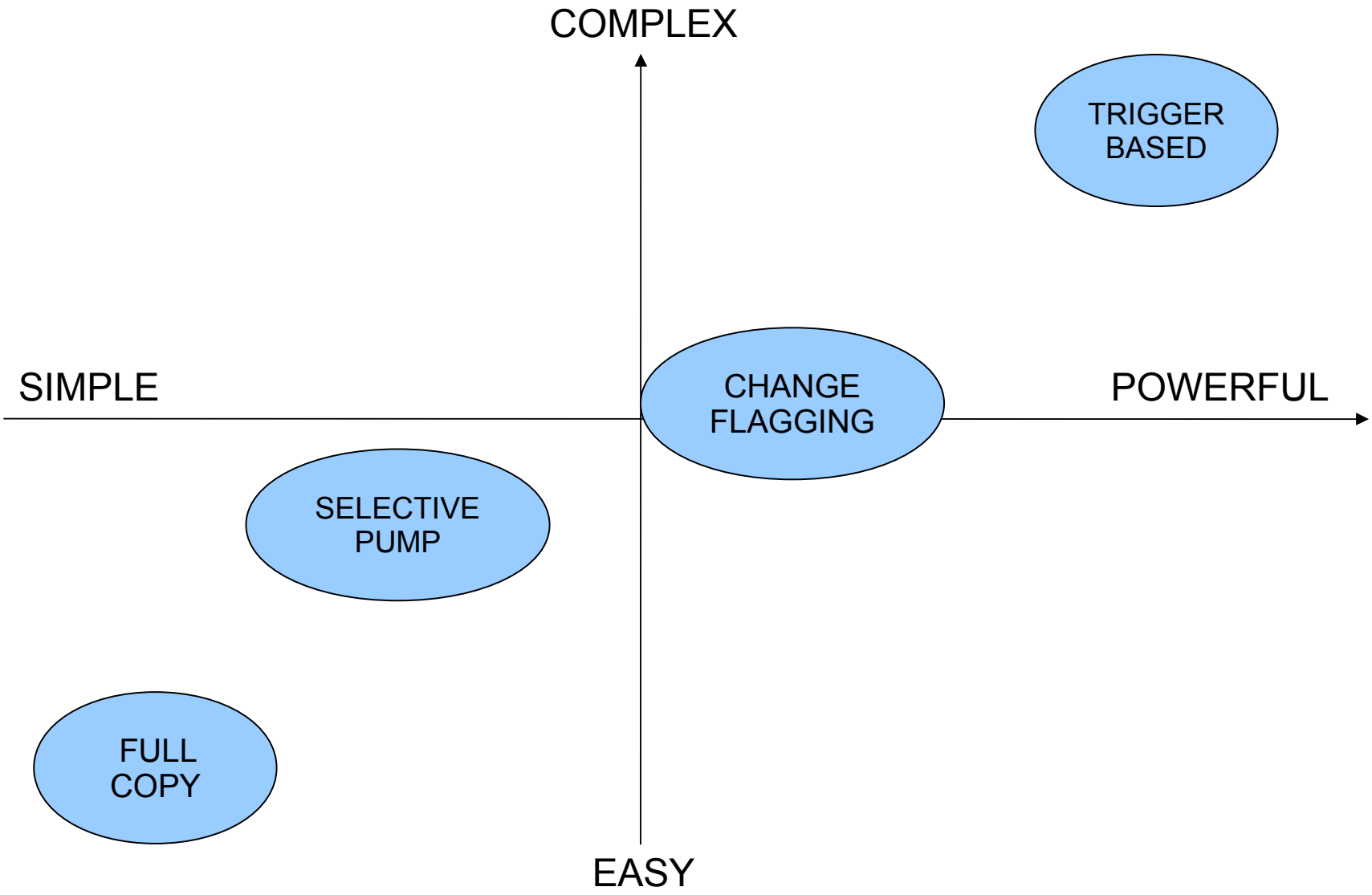
APPROACHES TO DATA SYNC

4) **Triggers**

- Triggers detect & log record changes
- Changes can be logged for multiple nodes
- Allows reliable conflict management (we know what changed)
- Database-level approach (multi-app)
- Complex to setup

APPROACHES TO DATA SYNC

ITDevCon



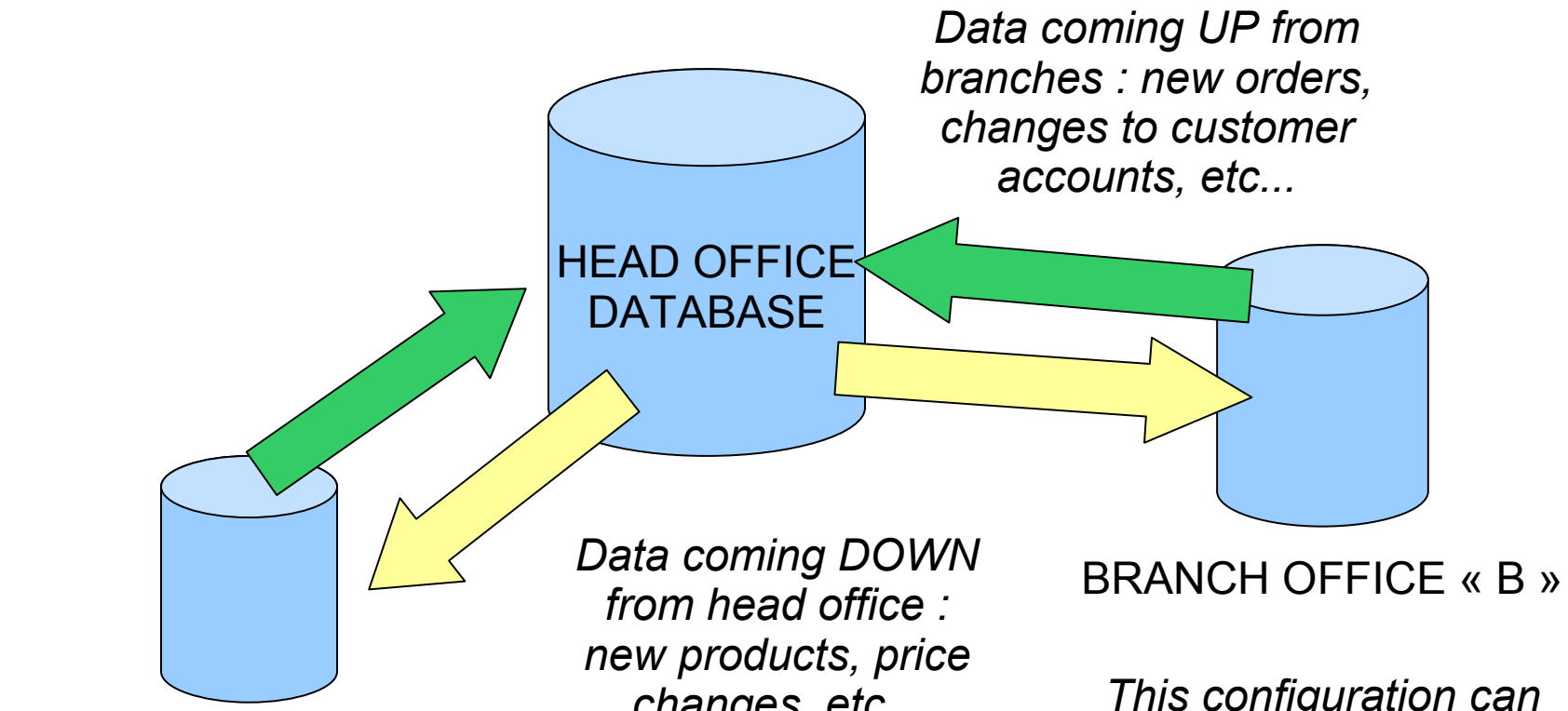
TRANSITION

We've seen the WAYS synchronization can be implemented...

...now let's see some real world situations that you will encounter where replication may be useful...

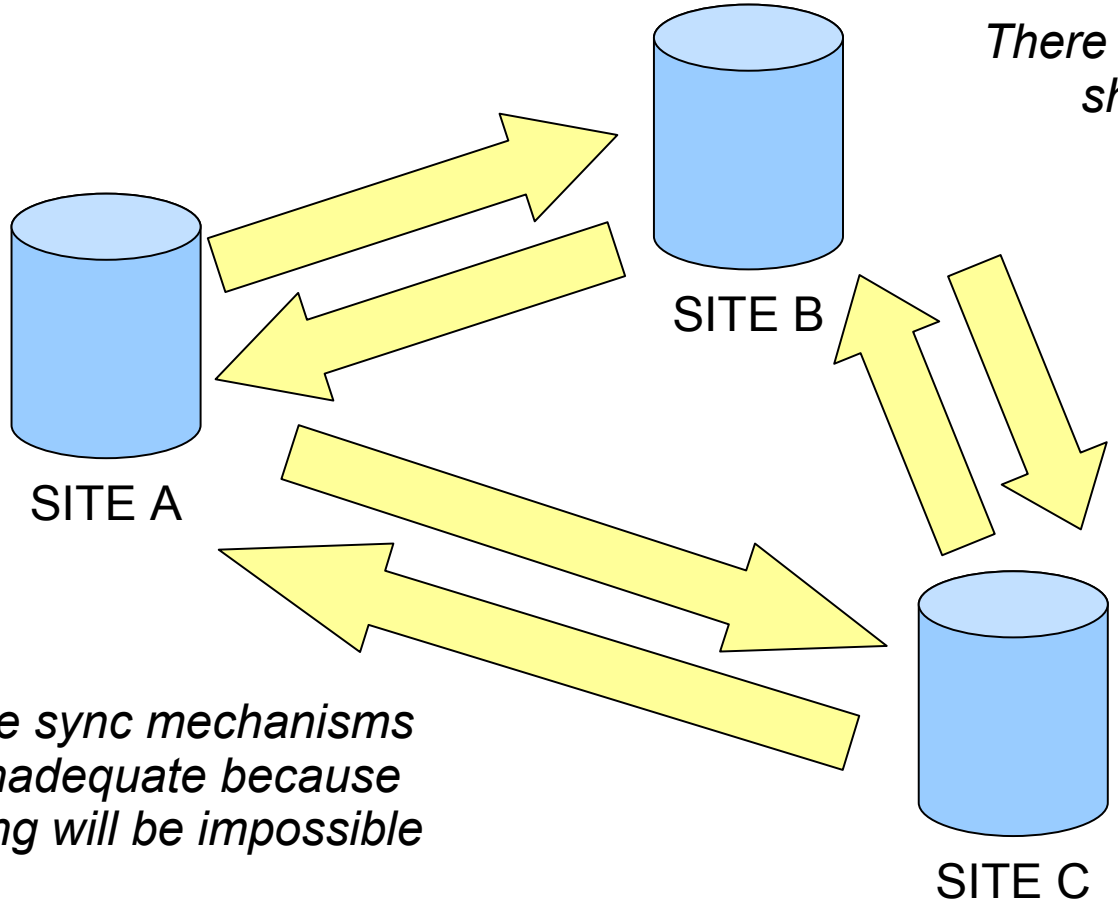
REAL-WORLD SITUATIONS

1) Head office / branch office



REAL-WORLD SITUATIONS

2) Multi-site distributed application



There are multiple sites that share all their data

Users must be able to edit the same data simultaneously on different sites

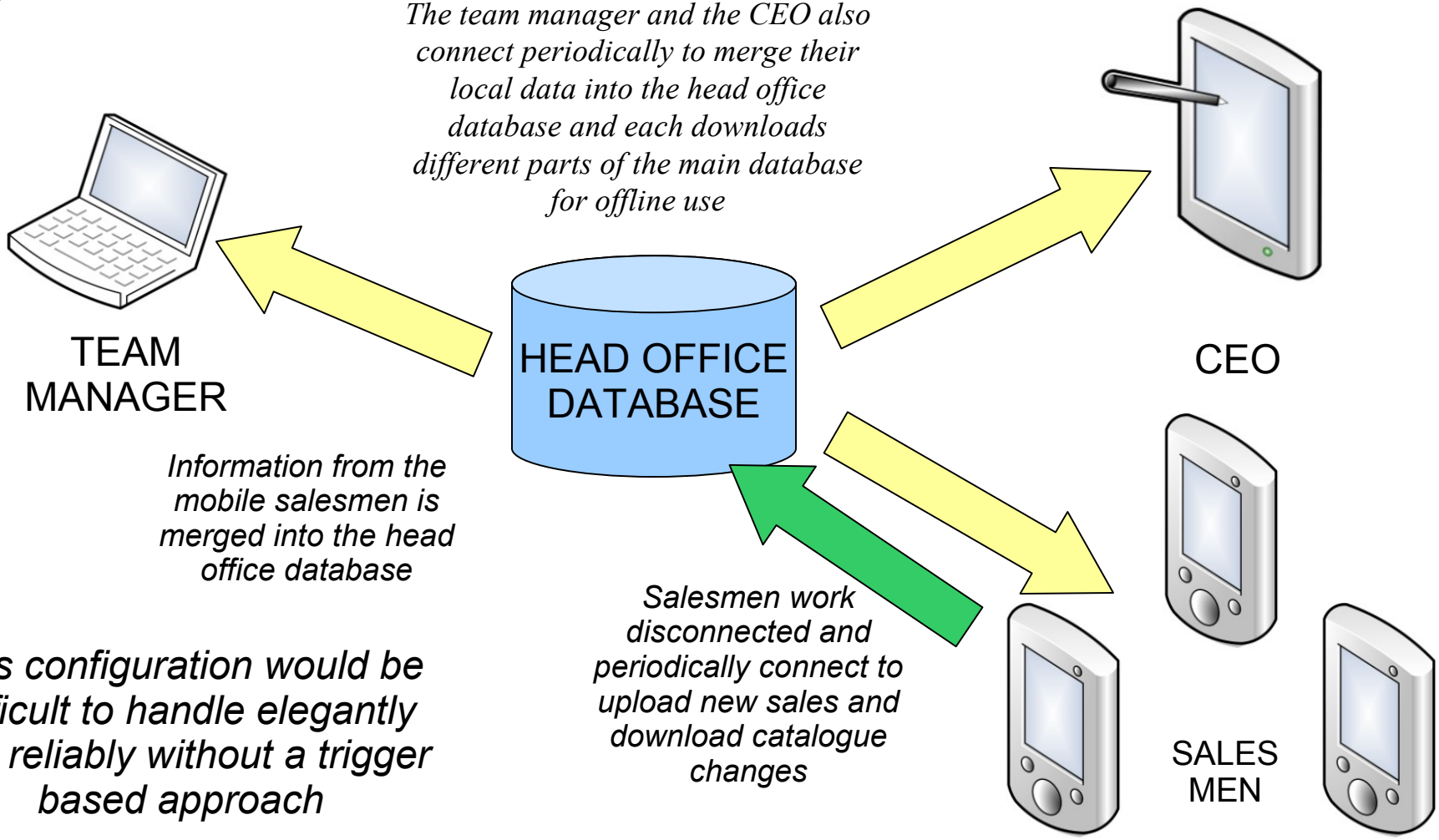
All databases need to be kept in sync at all times

Simple sync mechanisms are inadequate because merging will be impossible

REAL-WORLD SITUATIONS

3) Mobile users

The team manager and the CEO also connect periodically to merge their local data into the head office database and each downloads different parts of the main database for offline use



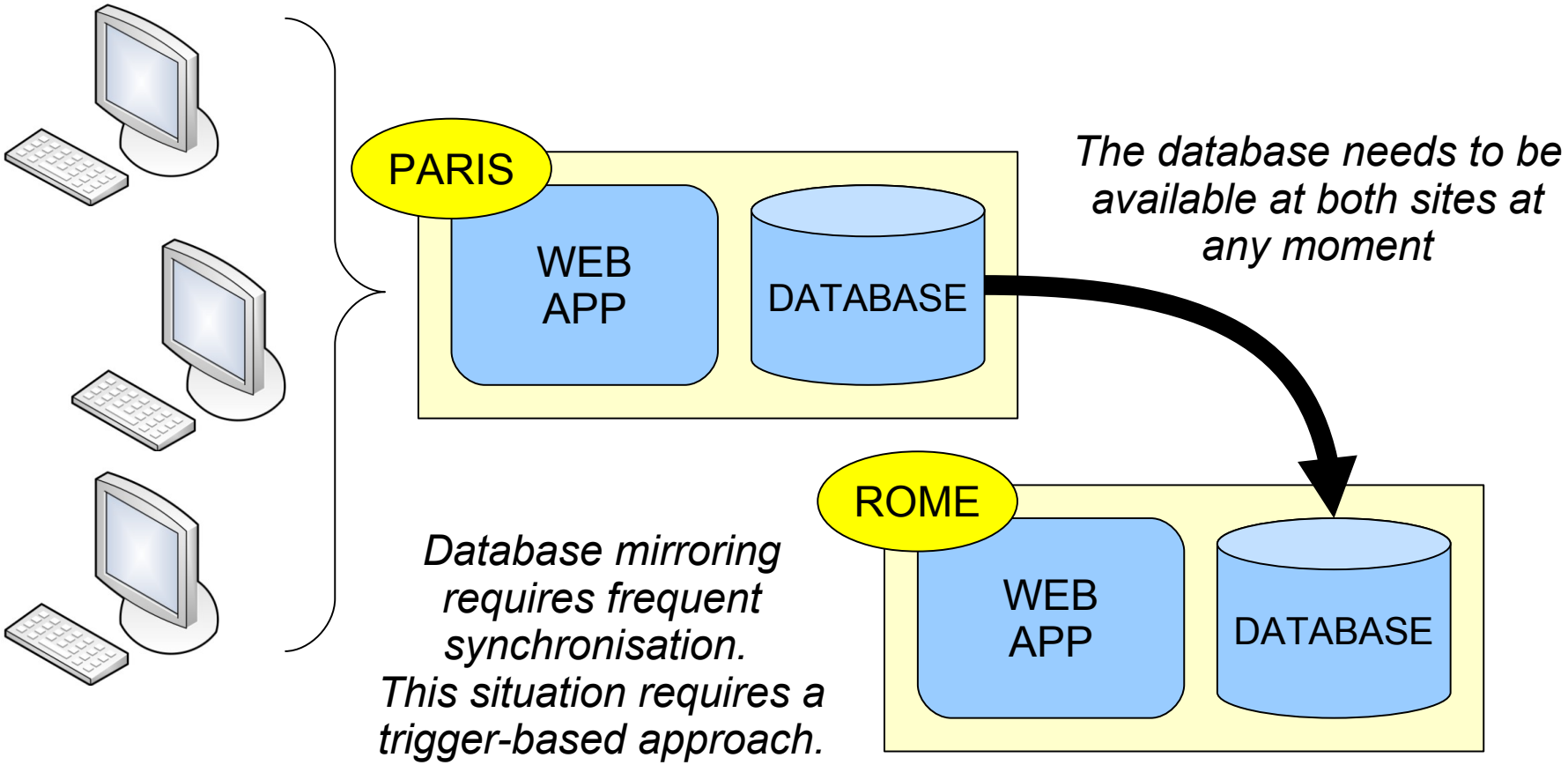
Information from the mobile salesmen is merged into the head office database

Salesmen work disconnected and periodically connect to upload new sales and download catalogue changes

This configuration would be difficult to handle elegantly and reliably without a trigger based approach

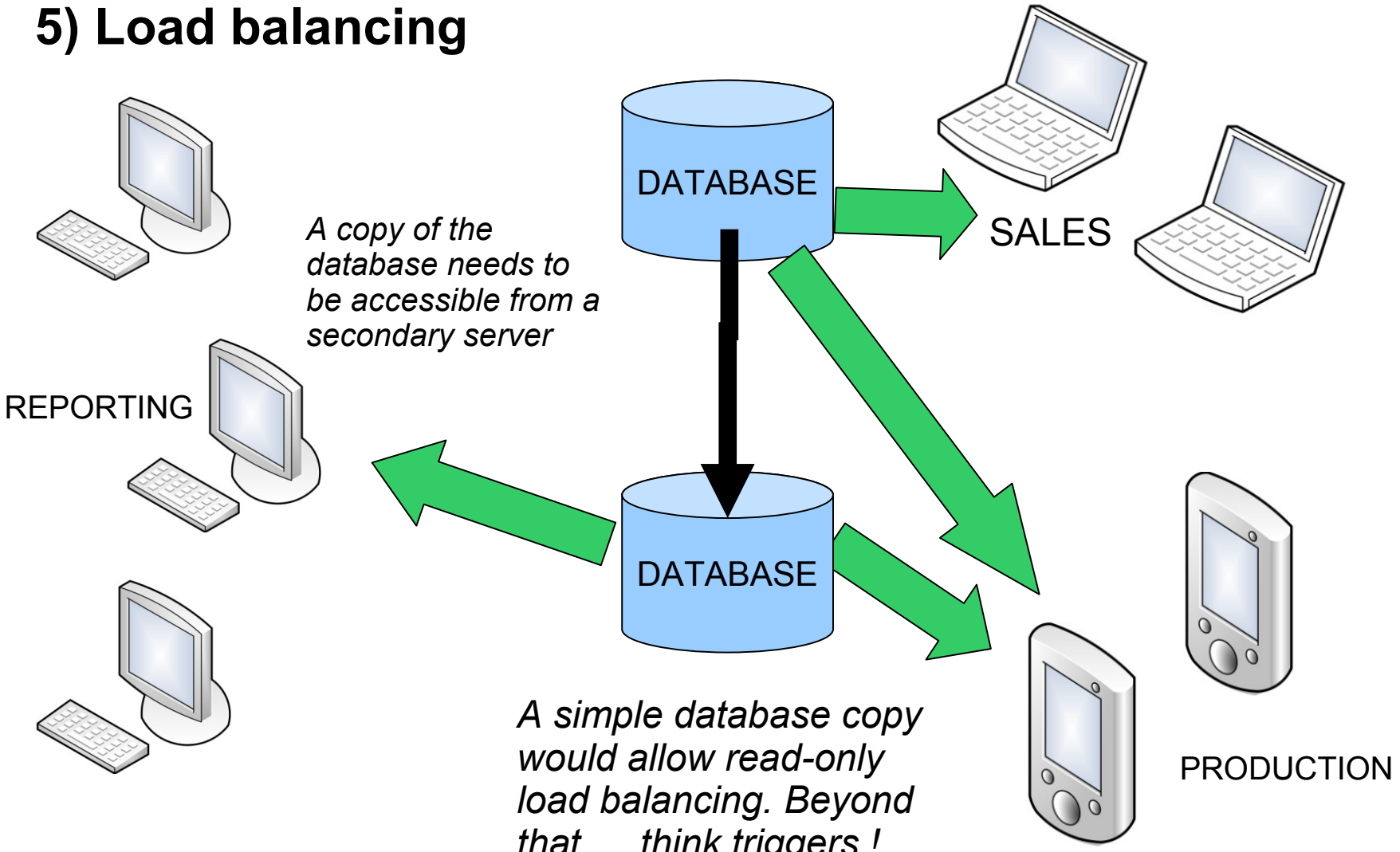
REAL-WORLD SITUATIONS

4) Live database mirroring (failover)



REAL-WORLD SITUATIONS

5) Load balancing



TIPS FOR IMPLEMENTING REPLICATION

- **Design your database with replication in mind**
  - Make reliable primary keys
    - Double PK (id + node name)
    - Single PK with prefix
    - GUIDs
  - Avoid summary tables :
    - E.g. : table with current stock per product
    - Instead use a detailed table with one row per change.
    - If need be, you can aggregate this data dynamically using a view or a stored procedure.
  - Beware of large update statements
    - Logging time (trigger execution time)
    - Performance issues (at replication time)

TIPS FOR IMPLEMENTING REPLICATION

- **Make small blocks of records**
  - Break up replication into small chunks to avoid needing to restart everything in case of broken connection.
  - Short-lived transactions avoid deadlocks between applications and replicator.

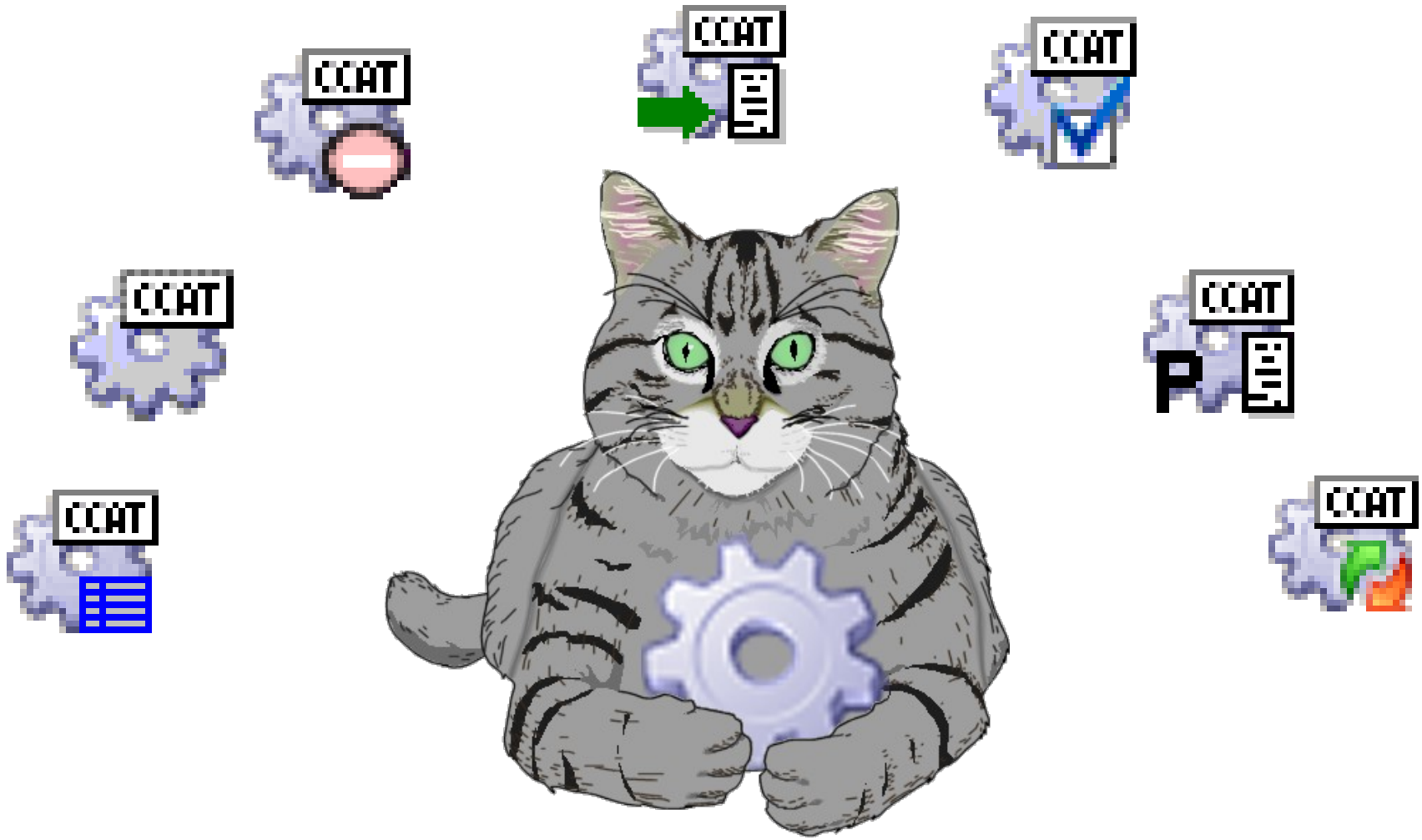
- **Replicate as often as possible**
  - Lower risk of conflicts.
  - Avoids ressource spikes.
  - Better user experience

- **Conflicts**
  - Set up automatic conflict resolution policies (if possible)
  - Bring remaining conflicts to user's attention immediately
  - Design applications so as to reduce possibility of conflicts

TIPS FOR IMPLEMENTING REPLICATION

- **Challenges setting up trigger-based replication**
  - Record bouncing
  - Design for resillience
    - Errors must be logged and failing statements retried
    - Databases may be inconsistent
      - » With one-way replication, destination database may have changed.
      - » Even with two-way replication, there may be inconsistency due to failed statements or administration mistakes
      - » Therefore, don't blindly replicate the same SQL operation (update/insert/delete) that occured in master database : check whether record exists or not, and update, insert or delete in order to make databases consistent.
    - Replication must be able to tolerate network connection loss

INTRODUCING COPYCAT



INTRODUCING COPYCAT

- Faced with these considerations, we developed a set of VCL components in order to encapsulate the complexities of trigger-based replication

- The components allow a clean separation between the business logic specific to each project and the low-level database replication mechanism

- This allows developers to « plug in » replication capabilities easily and reliably into their software

- The components take care of trigger creation, log parsing, conflict management and allow easy personalization via VCL events

HANDS-ON
DEMO !!